End-to-end optimization of goal-driven and visually grounded dialogue systems Harm de Vries

Florian Strub, Harm de Vries, Jeremie Mary, Bilal Piot, Aaron Courville, Olivier Pietquin

End-to-end optimization of goal-driven and visually grounded dialogue systems

Florian Strub
florian.strub@inria.fr
Univ. Lille, CNRS, Centrale Lille, Inria,
UMR 9189 - CRIStAL, F-59000 Lille, France

Harm de Vries
mail@harmdevries.com
University of Montreal

Jeremie Mary
jeremie.mary@univ-lille3.fr
Univ. Lille, CNRS, Centrale Lille, Inria,
UMR 9189 - CRIStAL, F-59000 Lille, France

Bilal Piot
piot@google.com
DeepMind

Aaron Courville
aaron.courville@gmail.com
University of Montreal

Olivier Pietquin
pietquin@google.com
DeepMind

Abstract

End-to-end design of dialogue systems has recently become a popular research topic thanks to powerful tools such as encoder-decoder architectures for sequence-to-sequence learning. Yet, most current approaches cast human-machine dialogue management as a supervised learning problem, aiming at predicting the next utterance of a participant given the full history of the dialogue. This vision is too simplistic to render the intrinsic planning problem inherent to dialogue as well as its grounded nature, making the context of a dialogue larger than the sole history. This is why only chit-chat and question answering tasks have been addressed so far using end-to-end architectures. In this paper, we introduce a Deep Reinforcement Learning method to optimize visually grounded task-oriented dialogues, based on the policy gradient algorithm. This approach is tested on a dataset of 120k dialogues collected through Mechanical Turk and provides encouraging results at solving both the problem of generating natural dialogues and the task of discovering a specific object in a complex picture.

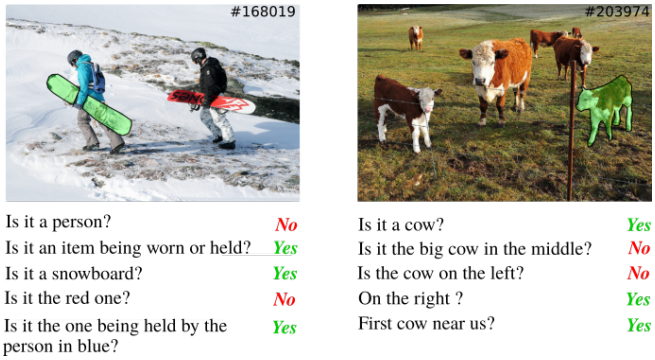


Figure 1: Two example games of the GuessWhat?! dataset. The correct object is highlighted by a green mask.

1 Introduction

Ever since the formulation of the Turing Test, building systems that can meaningfully converse with humans has been a long-standing goal of Artificial Intelligence (AI). Practical dialogue systems have to implement a management strategy that defines the system's behavior, for instance to decide when to provide information or to ask for clarification from the user. Although traditional approaches use linguistically motivated rules [Weizenbaum, 1966], recent methods are data-driven and make use of Reinforcement Learning (RL) [Lemon and Pietquin, 2007]. Significant progress in Natural Language Processing via Deep Neural Nets [Bengio *et al.*, 2003] made neural encoder-decoder architectures a promising way to train conversational agents [Vinyals and Le, 2015; Sordoni *et al.*, 2015; Serban *et al.*, 2016]. The main advantage of such end-to-end dialogue systems is that they make no assumption about the application domain and are simply trained in a supervised fashion from large text corpora [Lowe *et al.*, 2015].

However, there are many drawbacks to this approach. First, encoder-decoder models cast the dialogue problem into one of supervised learning, predicting the distribution over possible next utterances given the discourse so far. As with machine translation, this may result in inconsistent dialogues and errors that can accumulate over time. This is especially true because the action space of dialogue systems is vast, and existing datasets cover only a small subset of all trajectories, making it difficult to generalize to unseen scenarios [Mooney, 2006]. Second, the supervised learning framework does not account for the intrinsic planning problem that underlies dialogue, *i.e.* the sequential decision making process, which makes dialogue consistent over time. This is especially true when engaging in a task-oriented dialogue. As a consequence, reinforcement learning has been applied to dialogue systems since the late 90s [Levin *et al.*, 1997; Singh *et al.*, 1999] and dialogue optimization has been generally more studied than dialogue generation. Third, it doesn't naturally integrate external contexts (larger than the history of the dialogue) that is most often used by dialogue participants to interact. This context can be their physical environment,

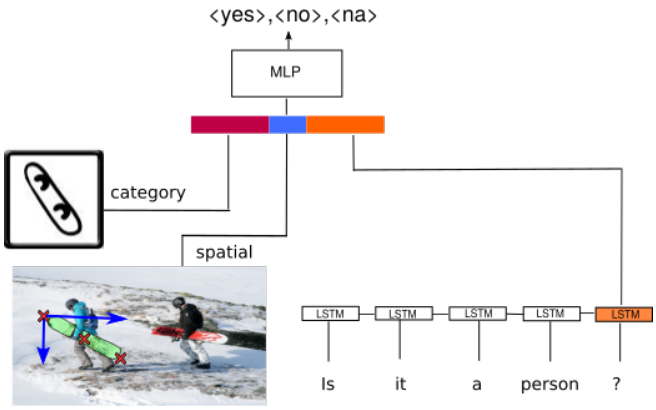


Figure 2: Oracle model.

a common task they try to achieve, a map on which they try to find their way, a database they want to access *etc*. It is part of the so called *Common Ground*, well studied in the discourse literature [Clark and Schaefer, 1989]. Over the last decades, the field of cognitive psychology has also brought empirical evidence that human representations are grounded in perception and motor systems [Barsalou, 2008]. These theories imply that a dialogue system should be grounded in a multi-modal environment in order to obtain human-level language understanding [Kiela *et al.*, 2016]. Finally, evaluating dialogues is difficult as there is not an automatic evaluation metric that correlates well with human evaluations [Liu *et al.*, 2016a].

On the other hand, RL approaches could handle the planning and the non-differentiable metric problems but require online learning (although batch learning is possible but difficult with low amounts of data [Pietquin *et al.*, 2011]). For that reason, user simulation has been proposed to explore dialogue strategies in a RL setting [Eckert *et al.*, 1997; Schatzmann *et al.*, 2006; Pietquin and Hastie, 2013]. It also requires the definition of an evaluation metric which is most often related to task completion and user satisfaction [Walker *et al.*, 1997]. In addition, successful applications of the RL framework to dialogue often rely on a predefined structure of the task, such as slot-filling tasks [Williams and Young, 2007] where the task can be casted as filling in a form.

In this paper, we present a global architecture for end-to-end RL optimization of a task-oriented dialogue system and its application to a multimodal task, grounding the dialogue into a visual context. To do so, we start from a corpus of 150k human-human dialogues collected via the recently introduced GuessWhat?! game [de Vries *et al.*, 2016]. The goal of the game is to locate an unknown object in a natural picture by asking a series of questions. This task is hard since it requires scene understanding and, more importantly, a dialogue strategy that leads to identify the object rapidly. Out of these data, we first build a supervised agent and a neural training environment. It serves to train a DeepRL agent online which is able to solve the task. We then quantitatively and qualitatively compare the performance of our system to a supervised approach on the same task from a human baseline perspective. In short, our contributions are:

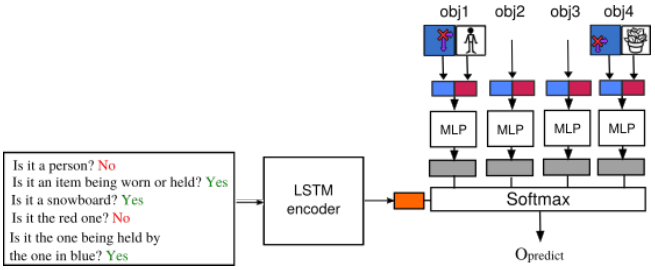


Figure 3: Guesser model.

- to propose the first multimodal goal-directed dialogue system optimized via Deep RL;
- to achieve 10% improvement on task completion over a supervised learning baseline.

2 GuessWhat?! game

We briefly explain here the GuessWhat?! game that will serve as a task for our dialogue system, but refer to [de Vries *et al.*, 2016] for more details regarding the task and the exact content of the dataset. It is composed of more than 150k human-human dialogues in natural language collected through Mechanical Turk.

2.1 Rules

GuessWhat?! is a cooperative two-player game in which both players see the picture of a rich visual scene with several objects. One player – the oracle – is randomly assigned an object (which could be a person) in the scene. This object is not known by the other player – the questioner – whose goal is to locate the hidden object. To do so, the questioner can ask a series of yes-no questions which are answered by the oracle as shown in Fig 1. Note that the questioner is not aware of the list of objects and can only see the whole picture. Once the questioner has gathered enough evidence to locate the object, he may choose to guess the object. The list of objects is revealed, and if the questioner picks the right object, the game is considered successful.

2.2 Notation

Before we proceed, we establish the GuessWhat?! notations that are used throughout the rest of this paper. A game is defined by a tuple $(\mathcal{I}, D, O, o^*)$ where $\mathcal{I} \in \mathbb{R}^{H \times W}$ is a picture of height H and width W, D a dialogue with J question-answer pairs $D = (\boldsymbol{q}_j, a_j)_{j=1}^{J}$, O a list of K objects $O = (o_k)_{k=1}^{K}$ and o^* the target object. Moreover, each question $\boldsymbol{q}_j = (w_i^j)_{i=1}^{I_j}$ is a sequence of length I_j with each token w_i^j taken from a predefined vocabulary V. The vocabulary V is composed of a predefined list of words, a question tag $<?>$ that ends a question and a stop token $<stop>$ that ends a dialogue. An answer is restricted to be either yes, no or not applicable *i.e.* $a_j \in \{<yes>, <no>, <na>\}$. For each object k, an object category $c_k \in \{1, \ldots, C\}$ and a pixel-wise segmentation mask $S_k \in \{0, 1\}^{H \times W}$ are available. Finally, to access subsets of a list, we use the following notations. If $l = (l_i^j)_{i=1}^{I,J}$ is a double-subscript list, then $l_{1:i}^j = (l_p^j)_{p=1}^{i,j}$ are

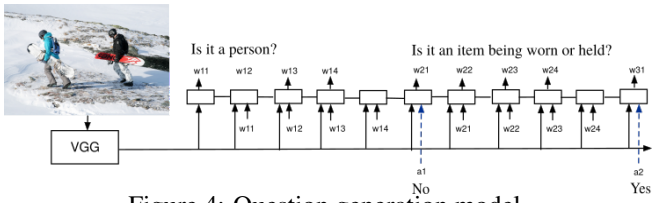


Figure 4: Question generation model.

the i first elements of the j^{th} list if $1 \leq i \leq I_j$, otherwise $l^j_{1:p} = \emptyset$. Thus, for instance, $w^j_{1:i}$ refers to the first i tokens of the j^{th} question and $(\boldsymbol{q}, a)_{1:j}$ refers to the j first question-answer pairs of a dialogue.

3 Training environment

From the GuessWhat?! dataset, we build a training environment that allows RL optimization of the questioner task by creating models for the oracle and guesser tasks. We also describe the supervised learning baseline to which we will compare. This mainly reproduces baselines introduced in [de Vries *et al.*, 2016].

Question generation baseline We split the questioner's job into two different tasks: one for asking the questions and another one for guessing the object. The question generation task requires to produce a new question $\boldsymbol{q}_{j+1}$, given an image $\mathcal{I}$ and a history of j questions and answers $(\boldsymbol{q}, a)_{1:j}$. We model the question generator (QGen) with a recurrent neural network (RNN), which produces a sequence of RNN state vectors $\boldsymbol{s}^j_{1:i}$ for a given input sequence $\boldsymbol{w}^j_{1:i}$ by applying the transition function f: $\boldsymbol{s}^j_{i+1} = f(\boldsymbol{s}^j_i, w^j_i)$. We use the popular long-short term memory (LSTM) cell [Hochreiter and Schmidhuber, 1997] as our transition function. In order to construct a probabilistic sequence model, one can add a softmax function g that computes a distribution over tokens w^j_i from vocabulary V. In the case of GuessWhat?!, this output distribution is conditioned on all previous questions and answers tokens as well as the image $\mathcal{I}$:

$$p(w^j_i | w^j_{1:i-1}, (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}).$$

We condition the model on the image by obtaining its VGG16 FC8 features and concatenating it to the input embedding at each step, as illustrated in Fig. 4. We train the model by minimizing the conditional negative log-likelihood:

$$-\log p(\boldsymbol{q}_{1:J} | a_{1:J}, \mathcal{I}) = -\log \prod_{j=1}^{J} p(\boldsymbol{q}_j | (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}),$$

$$= -\sum_{j=1}^{J} \sum_{i=1}^{I_j} \log p(w^j_i | w^j_{1:i-1}, (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}).$$

At test time, we can generate a sample $p(\boldsymbol{q}_j | (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I})$ from the model as follows. Starting from the state $\boldsymbol{s}^j_1$, we sample a new token w^j_i from the output distribution g and feed the embedded token $e(w^j_i)$ back as input to the RNN. We repeat this loop till we encounter an end-of-sequence token. To approximately find the most likely question, $\max_{\boldsymbol{q}_j} p(\boldsymbol{q}_j | (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I})$, we use the commonly used beam-search procedure. This heuristics aims to find the most likely sequence of words by exploring a subset of all questions and keeping the K-most promising candidate sequences at each time step.

Oracle The oracle task requires to produce a yes-no answer for any object within a picture given a natural language question. We outline here the neural network architecture that achieved the best performance and refer to [de Vries *et al.*, 2016] for a thorough investigation of the impact of other object and image information. First, we embed the spatial information of the crop by extracting an 8-dimensional vector of the location of the bounding box $[x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}]$ where w_{box} and h_{box} denote the width and height of the bounding box , respectively. We normalize the image height and width such that coordinates range from -1 to 1, and place the origin at the center of the image. Second, we convert the object category c^* into a dense category embedding using a learned look-up table. Finally, we use a LSTM to encode the current question q. We then concatenate all three embeddings into a single vector and feed it as input to a single hidden layer MLP that outputs the final answer distribution $p(a|q, c^*, x^*_{spatial})$ using a softmax layer, illustrated in Fig. 2.

Guesser The guesser model takes an image $\mathcal{I}$ and a sequence of questions and answers $(\boldsymbol{q}, a)_{1:N}$, and predicts the correct object o^* from the set of all objects. This model considers a dialogue as one flat sequence of question-answer tokens and use the last hidden state of the LSTM encoder as our dialogue representation. We perform a dot-product between this representation and the embedding for all the objects in the image, followed by a softmax to obtain a prediction distribution over the objects. The object embeddings are obtained from the categorical and spatial features. More precisely, we concatenate the 8-dimensional spatial representation and the object category look-up and pass it through an MLP layer to get an embedding for the object. Note that the MLP parameters are shared to handle the variable number of objects in the image. See Fig 3 for an overview of the guesser.

3.1 Generation of full games

With the question generation, oracle and guesser model we have all components to simulate a full game. Given an initial image $\mathcal{I}$, we generate a question $\boldsymbol{q}_1$ by sampling tokens from the question generation model until we reach the question-mark token. Alternatively, we can replace the sampling procedure by a beam-search to approximately find the most likely question according to the generator. The oracle then takes the question $\boldsymbol{q}_1$, the object category c^* and $x^*_{spatial}$ as inputs, and outputs the answer a_1. We append $(\boldsymbol{q}_1, a_1)$ to the dialogue and repeat generating question-answer pairs until the generator emits a stop-dialogue token or the maximum number of question-answers is reached. Finally, the guesser model takes the generated dialogue D and the list of objects O and predicts the correct object.

4 GuessWhat?! from RL perspective

One of the drawbacks of training the QGen in a supervised learning setup is that its sequence of questions is not explic-

itly optimized to find the correct object. Such training objectives miss the planning aspect underlying (goal-oriented) dialogues. In this paper, we propose to cast the question generation task as a RL task. More specifically, we use the training environment described before and consider the oracle and the guesser as part of the RL agent environment. In the following, we first formalize the GuessWhat?! task as a Markov Decision Process (MDP) so as to apply a policy gradient algorithm to the QGen problem.

4.1 GuessWhat?! as a Markov Decision Process

We define the state $\boldsymbol{x_t}$ as the status of the game at step t. Specifically, we define $\boldsymbol{x_t} = ((w_1^j, \dots, w_i^j), (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I})$ where $t = \sum_{j=1}^{j-1} I_j + i$ corresponds to the number of tokens generated since the beginning of the dialogue. An action u_t corresponds to select a new word w_{i+1}^j in the vocabulary V. The transition to the next state depends on the selected action:

- If $w_{i+1}^j = <stop>$, the full dialogue is terminated.
- If $w_{i+1}^j = <?>$, the ongoing question is terminated and an answer a_j is sampled from the oracle. The next state is $\boldsymbol{x}_{t+1} = ((\boldsymbol{q}, a)_{1:j}, \mathcal{I})$ where $\boldsymbol{q}_j = (w_1^j, \dots, w_i^j, <?>)$.
- Otherwise the new word is appended to the ongoing question and $\boldsymbol{x}_{t+1} = ((w_1^j, \dots, w_i^j, w_{i+1}^j), (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I})$.

Questions are automatically terminated after I_{max} words. Similarly, dialogues are terminated after J_{max} questions. Furthermore, a reward $r(\boldsymbol{x}, u)$ is defined for every state-action pair. A trajectory $\tau = (\boldsymbol{x}_t, u_t, \boldsymbol{x}_{t+1}, r(\boldsymbol{x}_t, u_t))_{1:T}$ is a finite sequence of tuples of length T which contains a state, an action, the next state and the reward where $T \leq J_{max} * I_{max}$. Thus, the game falls into the episodic RL scenario as the dialogue terminates after a finite sequence of question-answer pairs. Finally, the QGen output can be viewed as a stochastic policy $\pi_{\boldsymbol{\theta}}(u|\boldsymbol{x})$ parametrized by $\boldsymbol{\theta}$ which associates a probability distribution over the actions (i.e. words) for each state (*i.e.* intermediate dialogue and picture).

4.2 Training the QGen with Policy Gradient

While several approaches exist in the RL literature, we opt for policy gradient methods because they are known to scale well to large action spaces. This is especially important in our case because the vocabulary size is nearly 5k words. The goal of policy optimization is to find a policy $\pi_{\boldsymbol{\theta}}(u|\boldsymbol{x})$ that maximizes the expected return, also known as the mean value:

$$J(\boldsymbol{\theta}) = E_{\pi_{\boldsymbol{\theta}}}\big[\sum_{t=1}^{T} \gamma^{t-1} r(\boldsymbol{x}_t, u_t)\big],$$

where $\gamma \in [0, 1]$ is the discount factor, T the length of the trajectory and the starting state x_1 is drawn from a distribution p_1. Note that $\gamma = 1$ is allowed as we are in the episodic scenario [Sutton *et al.*, 1999]. To improve the policy, its parameters can be updated in the direction of the gradient of the mean value:

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \nabla_\theta J|_{\theta=\theta_h},$$

where h denotes the training time-step and α_h is a learning rate such that $\sum_{h=1}^{\infty} \alpha_h = \infty$ and $\sum_{h=1}^{\infty} \alpha_h^2 < \infty$.

Thanks to the gradient policy theorem [Sutton *et al.*, 1999], the gradient of the mean value can be estimated from a batch of trajectories $\mathcal{T}_h$ sampled from the current policy $\pi_{\boldsymbol{\theta}_h}$ by:

$$\nabla J(\boldsymbol{\theta}_h) = \Big\langle \sum_{t=1}^{T} \sum_{u_t \in V} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t)(Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b) \Big\rangle_{\mathcal{T}_h}, \tag{1}$$

where $Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$ is the state-action value function that estimates the cumulative expected reward for a given state-action couple and b some arbitrarily baseline function which can help reducing the variance of the estimation of the gradient. More precisely

$$Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) = E_{\pi_{\boldsymbol{\theta}}}\big[\sum_{t'=t}^{T} \gamma^{t'-t} r(\boldsymbol{x}_{t'}, u_{t'})\big].$$

Notice that the estimate in Eq (1) only holds if the probability distribution of the initial state $\boldsymbol{x_1}$ is uniformly distributed. The state-action value-function $Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$ can then be estimated by either learning a function approximator (Actor-critic methods) or by Monte-Carlo rollouts (REINFORCE [Williams, 1992]). In REINFORCE, the inner sum of actions is estimated by using the actions from the trajectory. Therefore, Eq (1) can be simplified to:

$$\nabla J(\boldsymbol{\theta}_h) = \Big\langle \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t)(Q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b) \Big\rangle_{\mathcal{T}_h}. \tag{2}$$

Finally, by using the GuessWhat?! game notation for Eq (2), the policy gradient for the QGen can be written as:

$$\nabla J(\boldsymbol{\theta}_h) = \Big\langle \sum_{j=1}^{J} \sum_{i=1}^{I_j} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(w_i^j | w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}) \\ (Q^{\pi_{\boldsymbol{\theta}_h}}((w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathcal{I}), w_i^j) - b) \Big\rangle_{\mathcal{T}_h}. \tag{3}$$

4.3 Reward Function

One tedious aspect of RL is to define a correct and valuable reward function. As the optimal policy is the result of the reward function, one must be careful to design a reward that would not change the expected final optimal policy [Ng *et al.*, 1999]. Therefore, we put a minimal amount of prior knowledge into the reward function and construct a zero-one reward depending on the guesser's prediction:

$$r(\boldsymbol{x}_t, u_t) = \begin{cases} 1 & \text{If } \text{argmax}_o[Guesser(\boldsymbol{x}_t)] = o* \text{ and } t = T \\ 0 & \text{Otherwise} \end{cases}.$$

So, we give a reward of one if the correct object is found from the generated questions, and zero otherwise.

Note that the reward function requires the target object $o*$ while it is not included in the state $\boldsymbol{x} = ((\boldsymbol{q}, a)_{1:J}, \mathcal{I})$. This breaks the MDP assumption that the reward should be a function of the current state and action. However, policy gradient methods, such as REINFORCE, are still applicable if the MDP is partially observable [Williams, 1992].

Algorithm 1 Training of QGen with REINFORCE

Require: Pretrained QGen,Oracle and Guesser
Require: Batch size K
for Each update **do**
  # Generate trajectories $\mathcal{T}_h$
  for $k = 1$ to K **do**
    Pick Image $\mathcal{I}_k$ and the target object $o_k^* \in O_k$
    # Generate question-answer pairs $(\boldsymbol{q}, a)_{1:j}^k$
    for $j = 1$ to J_{max} **do**
      $q_j^k = QGen(\boldsymbol{q}, a)_{1:j-1}^k, \mathcal{I}_k)$
      $a_j^k = Oracle(\boldsymbol{q}_j^k, o_k^*, \mathcal{I}_k)$
      if $<stop> \in \boldsymbol{q}_j^k$ **then**
        delete $(q, a)_j^k$ and break;
    $p(o_k|\cdot) = Guesser((q, a)_{1:j}^k, \mathcal{I}_k, O_k)$
    $r(\boldsymbol{x}_t, u_t) = \begin{cases} 1 & \text{If argmax}_{o_k} p(o_k|\cdot) = o_k^* \\ 0 & \text{Otherwise} \end{cases}$
  Define $\mathcal{T}_h = ((q, a)_{1:j_k}^k, \mathcal{I}_k, r_k)_{1:K}$
  Evaluate $\nabla J(\boldsymbol{\theta}_h)$ with Eq. (3) with $\mathcal{T}_h$
  SGD update of QGen parameters $\boldsymbol{\theta}$ using $\nabla J(\boldsymbol{\theta}_h)$
  Evaluate $\nabla L(\boldsymbol{\phi}_h)$ with Eq. (4) with $\mathcal{T}_h$
  SGD update of baseline parameters using $\nabla L(\boldsymbol{\phi}_h)$

4.4 Full training procedure

For the QGen, oracle and guesser, we use the model architectures outlined in section 3. We first independently train the three models with a cross-entropy loss. We then keep the oracle and guesser models fixed, while we train the QGen in the described RL framework. It is important to pretrain the QGen to kick-start training from a reasonable policy. The size of the action space is simply too big to start from a random policy.

In order to reduce the variance of the policy gradient, we implement the baseline $b_{\boldsymbol{\phi}}(\boldsymbol{x}_t)$ as a function of the current state, parameterized by $\boldsymbol{\phi}$. Specifically, we use a one layer MLP which takes the LSTM hidden state of the QGen and predicts the expected reward. We train the baseline function by minimizing the Mean Squared Error (MSE) between the predicted reward and the discounted reward of the trajectory at the current time step:

$$L(\boldsymbol{\phi_h}) = \left\langle \left[b_{\phi_h}(\boldsymbol{x}_t) - \sum_{t'=t}^{T} \gamma^{t'} r_{t'} \right]^2 \right\rangle_{\mathcal{T}_h} \quad (4)$$

We summarize our training procedure in Algorithm 1.

5 Related work

Outside of the dialogue literature, RL methods have been applied to encoder-decoder architectures in machine translation [Ranzato *et al.*, 2016; Bahdanau *et al.*, 2017] and image captioning [Liu *et al.*, 2016b]. In those scenarios, the BLEU score is used as a reward signal to fine-tune a network trained with a cross-entropy loss. However, the BLEU score is a surrogate for human evaluation of naturalness, so directly optimizing this measure does not guarantee improvement in the translation/captioning quality. In contrast, our reward function encodes task completion, and optimizing this metric is exactly what we aim for. Finally, the BLEU score can only be used in a batch setting because it requires the ground-truth labels from the dataset. In GuessWhat?!, the computed reward is independent from the *human* generated dialogue.

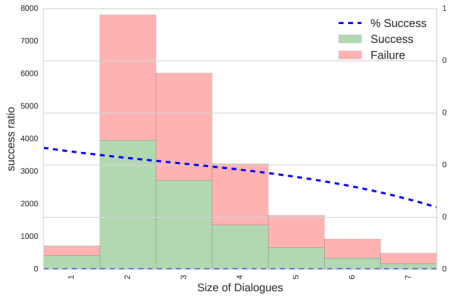


Figure 5: Task completion ratio of REINFORCE trained QGEN for given dialogue length.

Although visually-grounded language models have been studied for a long time [Roy, 2002], important breakthroughs in both visual and natural language understanding has led to a renewed interest in the field [LeCun *et al.*, 2015]. Especially image captioning [Lin *et al.*, 2014] and visual question answering [Antol *et al.*, 2015] has received much attention over the last few years, and encoder-decoder models [Liu *et al.*, 2016b; Lu *et al.*, 2016] have shown promising results for these tasks. Only very recently the language grounding tasks have been extended to a dialogue setting with the Visual Dialog [Das *et al.*, 2016] and GuessWhat?! [de Vries *et al.*, 2016] datasets. While Visual Dialog considers the chit-chat setting, the GuessWhat?! game is goal-oriented which allows us to cast it in into an RL framework.

6 Experiments

As already said, we used the GuessWhat?! dataset[1] that includes 155,281 dialogues containing 821,955 question/answer pairs composed of 4900 words on 66,537 unique images and 134,074 unique objects. The experiments source code is available at `https://guesswhat.ai`.

6.1 Training details

We pre-train the networks described in Section 3. After training, the oracle network obtains 21.5% error and the guesser network reports 36.2% error on the test set. Throughout the rest of this section we refer to the pretrained QGen as our baseline model. We then initialize our environment with the pre-trained models and train the QGen with REINFORCE for 80 epochs with plain stochastic gradient descent (SGD) with a learning rate of 0.001 and a batch size of 64. For each epoch, we sample each training images once, and randomly choose one of its object as the target. We simultaneously optimize the baseline parameters $\boldsymbol{\phi}$ with SGD with a learning rate of 0.001. Finally, we set the maximum number of questions to 8 and the maximum number of words to 12

6.2 Results

Accuracy Since we are interested in human-level performance, we report the accuracies of the models as a percentage

[1] Available at `https://guesswhat.ai/download`

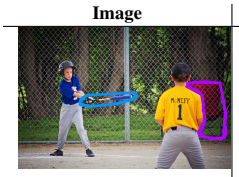
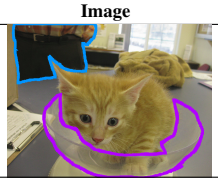
Image	Beam Search	REINFORCE	Image	Beam Search	REINFORCE
	Is it a person ? no Is it a ball ? no Is it a ball ? no Is it a ball ? no Is it a ball ? no Failure (blue bat)	Is it a person ? no Is a glove ? no Is an umbrella ? no Is in the middle ? no On a person? no is it on on far right? yes Success (red chair)		Is it a cat ? no Is it a book ? no Is it a book ? no Is it a book ? no Is it a book ? no Failure (person)	Is it a cat ? no Is it on the table ? yes Is it the book ? no Is it fully visible? yes Success (bowl)
	Is it a person ? yes Is it the one in front ? yes Is it the one on the left ? no Is it the one in the middle with the red umbrella ? yes Is it the one to the right of the girl in ? no Failure (umbrella)	Is it a person ? yes Is it girl in white ? yes Success (girl)		Is it a bag ? yes Is it red ? no Is it the one in the middle ? no Is it the one on the far right ? no Is it the one with the blue bag ? yes Success (most left bag)	Is it a suitcase? yes Is it in the left side ? yes Failure (left bag)

Table 1: Samples extracted from the test set. The blue (resp. purple) box corresponds to the object picked by the guesser for the beam-search (resp. REINFORCE) dialogue. The small verbose description is added to refer to the object picked by the guesser.

		New Objects	New Pictures
Baseline	Sampling	46.4% ± 0.2	45.0% ± 0.1
	Greedy	48.2% ± 0.1	46.9%
	BSearch	53.4% ± 0.0	53.0%
REINFORCE	Sampling	**63.2% ± 0.3**	**62.0% ± 0.2**
	Greedy	58.6% ± 0.0	57.5%
	BSearch	54.3% ± 0.1	53.2%

Table 2: Accuracies of the models as a percentage of human performance of the QGen trained with the baseline and REINFORCE. New objects refers to uniformly sampling objects within the training set, while new pictures refer to the test set.

of human performance (84.4%), estimated from the dataset. We report the scores in Table 2, in which we compare sampling objects from the training set (New Objects) and test set (New Pictures) i.e. unseen pictures. We report the standard deviation over 5 runs in order to account for the sampling stochasticity. On the test set, the baseline obtains 45.0% accuracy, while training with REINFORCE improves to 62.0%. This is also a significant improvement over the beam-search baseline, which achieves 53.0% on the test-set. The beam-search procedure improves over sampling from the baseline, but interestingly lowers the score for REINFORCE.

Samples We qualitatively compare the two methods by analyzing a few generated samples, as shown in Table 1. We observe that the beam-search baseline trained in a supervised fashion keeps repeating the same questions, as can be seen in the two top examples in Tab. 1. We noticed this behavior especially on the test set *i.e.* when confronted with unseen pictures, which may highlight some generalization issues. We also find that the beam-search baseline generates longer questions (7.1 tokens on average) compared to REINFORCE (4.0 tokens on average). This qualitative difference is clearly visible in the bottom-left example, which also highlights that the supervised baseline sometimes generates visually relevant but incoherent sequences of questions. For instance, asking "Is it the one to the right of the girl in?" is not a very logical follow-up of "Is it the one in the middle with the red umbrella?". In contrast, REINFORCE seem to implement a more grounded and relevant strategy: "Is it girl in white?" is a reasonable follow-up to "Is it a person?". In general, we observe that REINFORCE favor enumerating object categories ("is it a person?") or absolute spatial information ("Is it left?"). Note these are also the type of questions that the oracle is expected to answer correctly, hence, REINFORCE is able to tailor its strategy towards the strengths of the oracle.

Dialogue length For the REINFORCE trained QGen, we investigate the impact of the dialogue length on the success ratio in Fig. 5. Interestingly, REINFORCE learns to stop on average after 4.1 questions, although we did not encode a question penalty into the reward function. This policy may be enforced by the guesser since asking additional but noisy questions greatly lower the prediction accuracy of the guesser as shown in Tab. 1. Therefore, the QGen learns to stop asking questions when a dialogue contains enough information to retrieve the target object. However, we observe that the QGen sometimes stops too early, especially when the image contains too many objects of the same category. Interestingly, we also found that the beam-search fails to stop the dialogue. Beam-search uses a length-normalized log-likelihood to score candidate sequences to avoid a bias towards shorter questions. However, questions in GuessWhat?! almost always start with "is it", which increases the average log likelihood of a question significantly. The score of a new question might thus (almost) always be higher than emitting a single $<stop>$ token. Our finding was further confirmed by the fact that a sampling procedure did stop the dialogue.

Vocabulary Sampling from the supervised baseline on the test set results in 2,893 unique words, while sampling from the REINFORCE trained model reduces its size to 1,194. However, beam search only uses 512 unique words which is consistent with the observed poor variety of questions.

7 Conclusion

In this paper, we proposed to build a training environment from supervised deep learning baselines in order to train a DeepRL agent to solve a goal-oriented multi-modal dialogue task. We show the promise of this approach on the GuessWhat?! dataset, and observe quantitatively and qualitatively an encouraging improvement over a supervised baseline model. While supervised learning models fail to generate a coherent dialogue strategy, our method learns when to stop after generating a sequence of relevant questions.

Acknowledgement The authors would like to acknowledge the stimulating environment provided by the SequeL labs. We acknowledge the following agencies for research funding and computing support: CHISTERA IGLU and CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR.

References

[Antol *et al.*, 2015] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, Z. Lawrence, and D. Parikh. Vqa: Visual question answering. In *Proc. of ICCV*, 2015.

[Bahdanau *et al.*, 2017] D. Bahdanau, P. Brakel, K. Kelvin, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *Proc. of ICLR*, 2017.

[Barsalou, 2008] Lawrence W Barsalou. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645, 2008.

[Bengio *et al.*, 2003] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *JMLR*, 3(Feb):1137–1155, 2003.

[Clark and Schaefer, 1989] H. Clark and E. Schaefer. Contributing to discourse. *Cognitive Science*, 13(2):259–294, 1989.

[Das *et al.*, 2016] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. Moura, D. Parikh, and D. Batra. Visual Dialog. *arXiv preprint arXiv:1611.08669*, 2016.

[de Vries *et al.*, 2016] H. de Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville. GuessWhat?! Visual object discovery through multi-modal dialogue. *Proc. of CVPR*, 2016.

[Eckert *et al.*, 1997] W. Eckert, E. Levin, and R. Pieraccini. User modeling for spoken dialogue system evaluation. In *Proc. of ASRU*, 1997.

[Hochreiter and Schmidhuber, 1997] S. Hochreiter and J. Schmidhuber. Long short-term memory. volume 9, pages 1735–1780. MIT Press, 1997.

[Kiela *et al.*, 2016] D. Kiela, L. Bulat, A.Vero, and S. Clark. Virtual Embodiment: A Scalable Long-Term Strategy for Artificial Intelligence Research. *NIPS workshop in Machine Intelligence*, 2016.

[LeCun *et al.*, 2015] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553), 2015.

[Lemon and Pietquin, 2007] O. Lemon and O. Pietquin. Machine learning for spoken dialogue systems. In *Proc. of Interspeech*, 2007.

[Levin *et al.*, 1997] E. Levin, R. Pieraccini, and W. Eckert. Learning dialogue strategies within the markov decision process framework. In *Proc. of ASRU*, 1997.

[Lin *et al.*, 2014] TY. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and L. Zitnick. Microsoft coco: Common objects in context. In *Proc. of ECCV*, 2014.

[Liu *et al.*, 2016a] C. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proc. of EMNLP*, 2016.

[Liu *et al.*, 2016b] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy. Optimization of image description metrics using policy gradient methods. *Under review at CVPR*, 2016.

[Lowe *et al.*, 2015] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proc. of SIGdial*, 2015.

[Lu *et al.*, 2016] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical Question-Image Co-Attention for Visual Question Answering. In *Proc. of NIPS*, 2016.

[Mooney, 2006] R. Mooney. Learning language from perceptual context: A challenge problem for AI. In *Proc. of the 2006 AAAI Fellows Symposium*, 2006.

[Ng *et al.*, 1999] A. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. of ICML*, 1999.

[Pietquin and Hastie, 2013] Olivier Pietquin and Helen Hastie. A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*, 28(1):59–73, 003 2013.

[Pietquin *et al.*, 2011] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):7, 2011.

[Ranzato *et al.*, 2016] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *Proc. of ICLR*, 2016.

[Roy, 2002] D. Roy. Learning visually grounded words and syntax for a scene description task. *Computer speech & language*, 16(3):353–385, 2002.

[Schatzmann *et al.*, 2006] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126, 2006.

[Serban *et al.*, 2016] I. Serban, R. Lowe, L. Charlin, and J. Pineau. Generative Deep Neural Networks for Dialogue: A Short Review. *NIPS workshop Learning Methods for Dialogue*, 2016.

[Singh *et al.*, 1999] S. Singh, M. Kearns, D. Litman, and M. Walker. Reinforcement Learning for Spoken Dialogue Systems. In *Proc. of NIPS*, 1999.

[Sordoni *et al.*, 2015] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, JY. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proc. of NAACL HLT*, 2015.

[Sutton *et al.*, 1999] R. Sutton, D. McAllester, S. Singh, Y. Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *Proc. of NIPS*, 1999.

[Vinyals and Le, 2015] O. Vinyals and Q. Le. A neural conversational model. *ICML Deep Learning Workshop*, 2015.

[Walker *et al.*, 1997] M. Walker, D. Litman, C. Kamm, and A. Abella. Paradise: A framework for evaluating spoken dialogue agents. In *Proc. of ACL*, 1997.

[Weizenbaum, 1966] J. Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, 1966.

[Williams and Young, 2007] J. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[Williams, 1992] R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.